SOA software™

THE BEST-RUN BUSINESSES RUN SAP

SOA Management with Integrated solution from SAP and SOA Software

A report from SAP Co-innovation Lab

SOA Software:

Alistair Farquharson, CTO

Cherry Rose, Principal Consultant

Ian Goldsmith – Product Manager

SAP Co-Innovation Lab:

Canyang Kevin Liu,

Joerg Nalik,

Siva Gopal Modadugula

November 2008	1.0	

Table of Contents

1 Introduction

In today's Web-driven environment, traditional enterprise boundaries are being fundamentally reshaped. Enterprises can no longer survive just as largely self-contained silos. The success of an enterprise depends more and more on how well they are integrated with their business networks where hundreds of customers and partners each may require a unique relationship. Enterprise SOA adoption is a critical part of business network transformation and provides a more flexible and agile approach for enterprises to offer specialized business services to interweaved business networks. A successful SOA adoption brings enterprises new levels of competitive differentiations and new revenue opportunities.

From IT perspective, Web services technologies form the underpinning of SOA adoption, and as the usage of Web services increases, more and more sensitive business data may flow in and out the web and critical operational and performance problems can arise. SOA management addresses the management challenges introduced by SOA. It provides the capabilities to safeguard, operate, and evolve an enterprise's SOA runtime production environment. Comparing to SOA design and development time governance which helps enterprises to create right services in the right way, SOA management is more focus on making sure the services, once in production, are operated appropriately with right level of visibility and control. The term SOA management is also known as SOA operational governance. In this paper, we use these two terms interchangeably.

SOA Management is best viewed as a solution, not a product by itself, and the overall solution may utilize the capabilities of several infrastructure products from multiple vendors to synergistically address the SOA management challenges. SAP seeks to offer its customers the very best SOA management solutions. With thousands of packaged high quality enterprise services delivered from SAP, customers can immediately reap the benefits of SAP's proven design time governance that focuses on building alignment of SOA infrastructure with the key business processes. Once customers decided to put these enterprise services into production use in their own SOA landscape, they can leverage SAP Solution Manager and SAP NetWeaver Administrator as the cornerstones for runtime management of these enterprise services. SAP Solution Manager is an industry standard ITIL based solution that provides first class application management capabilities, including growing features for managing the challenges and opportunities of SOA landscapes. With a consistent management approach for SOA and non-SOA components in a customer's landscape, SAP Solution Manager ensures low TCO based on proven IT management best practices.

In addition to the SOA management capabilities provided in SAP Solution Manager and SAP NetWeaver Administrator, SAP works closely via its Co-Innovation Lab (COIL) with its customers and leading SOA management vendors in defining integrated innovative solutions to address management challenge of customers' highly heterogeneous SOA landscape. COIL is one of the newest communities of innovation that make up the SAP ecosystem. It serves as a hands-on working environment for SAP, its customers, and partners to execute joint projects, and work on proof of concepts, enabling them to discover and promote new business applications and technology solutions. Via the best practices and integrated solutions for SOA management developed in COIL, customers benefit from reduced integration costs and faster innovation cycles while gaining a new generation of highly agile and adaptable solutions. The cooperation between SAP and leading SOA management vendors gives customers the freedom to choose which solution fits best with their infrastructure.

This paper provides a summary of the integrated SOA management solution developed in COIL by SAP and SOA Software. Headquartered in Los Angeles, California, SOA Software™ offers integrated SOA governance automation solution addressing SOA security and management together with legacy and B2B Web services requirements. SOA Software's offers a range of products that addresses different aspects of SOA governance. In the COIL project reported in this paper, we only focused on the SOA software products that can be easily ramped up into an SAP landscape to manage SAP enterprise services without changing SAP components. Other components may be considered for future COIL projects.

2 Integration of SOA Software components into COIL landscape

Every customer may have a different landscape, and it would not be possible to replicate each and every one of them in the lab. At COIL, we began with the basics by building a simplified SAP landscape which still includes the key elements of a modern data center and should be able to provide a common pattern for SOA Governance solution integration. More complexity may be added in future phases of the project.

2.1 The baseline COIL Landscape

The COIL landscape is described in Figure 1, and has the following characteristics:

- It includes a SAP Portal 7.0 server, a Composition Environment 7.10 SP3 Server, and an emulated ECC 6.0 backend which provides simulated SAP enterprise services with CE 7.10
- All application instances are deployed inside VMware virtual machines.
- All applications are deployed as 2 instances for basic high availability and scalability demonstration
- A commercial Application Delivery (AD) appliance is added which provides the following services:
    - Providing a virtual server proxy endpoint for the two instance of an application component and load balanced routing of network traffic to the component instances.
    - Termination of incoming SSL connection and decryption of such traffic
    - Optionally applying routing rules and transformation operations on traffic content
    - Connection multiplexing of many incoming TCP/IP connections to fewer connections towards the server side.
    - Optional re-encryption of traffic towards the application components

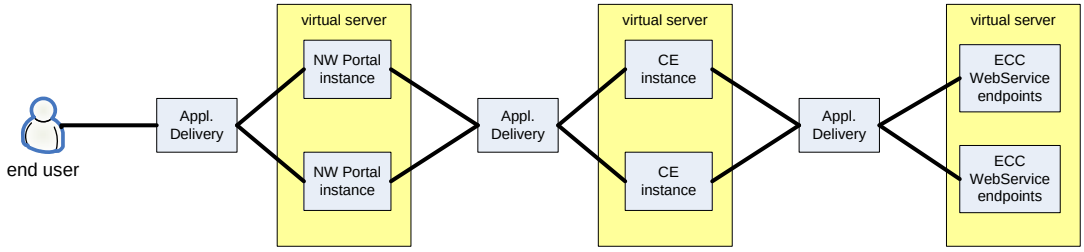


Figure 1 - COIL baseline landscape

On top of the above production-like landscape, COIL also provides a simple test application which leverages all the key components above. As shown in Figure 2, the application addresses a Customer Fact Sheet (CFS) scenario. It allows a sales person to log in to the Portal and look up customers. Clicking on one customer displays a quote and order history list for that customer. Behind the scenes, web service calls from the Portal to a composite application running on CE are triggered, and in turn the composite application triggers further web service calls to a backend system where the customer data resides.

In the first step, the user enters a customer name to search for a customer, and a list of matching customers are returned. Clicking on a line item in the returned customer list gets more details about the customer and the history of quotes and orders submitted by the selected customer.

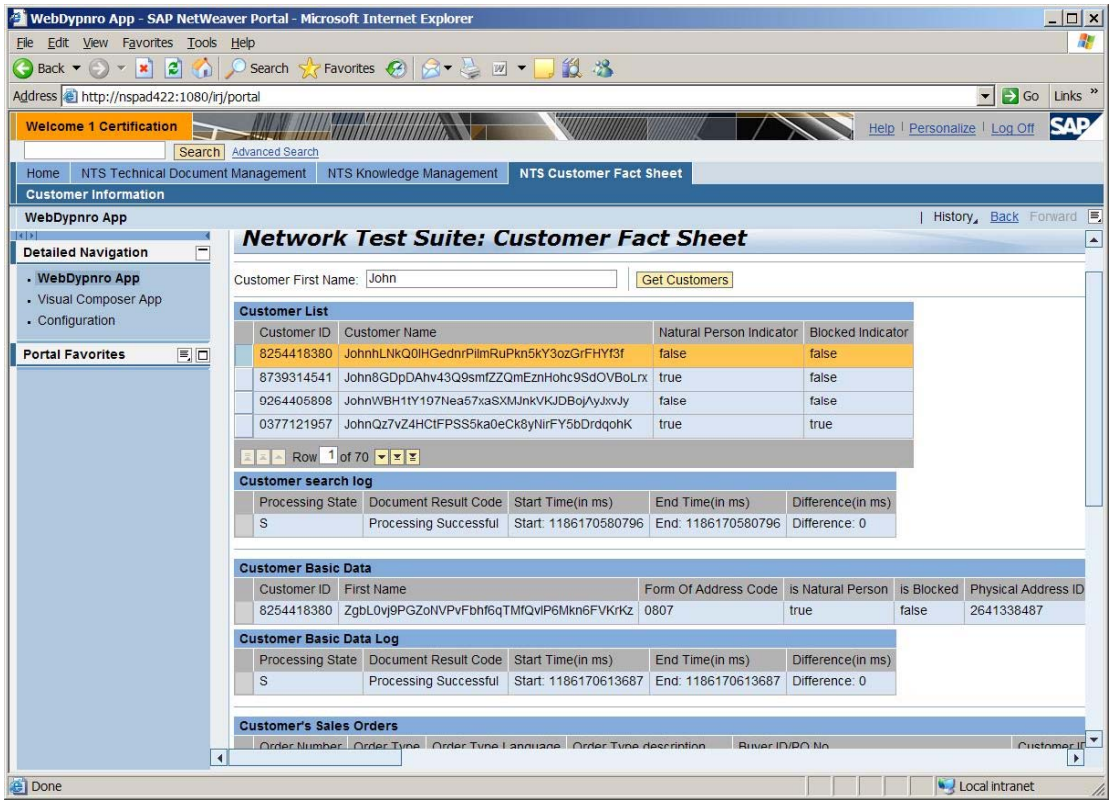


Figure 2 - COIL Customer Fact Sheet application

The whole scenario can be simulated using load testing tools. COIL we uses HP Load Runner for performance testing.

2.2 Key SOA Software components

As described in Figure 3, SOA Software's SOA operational governance solution consists of 2 key products: a **Service Manager** component that provides a set of distributed intermediaries for runtime Web services policy enforcement, metric collection, and audit; and a central **Policy Manager** component which provides a centralized toolset for policy development, configuration, monitoring, reporting, and other administrative tasks. These products integrate seamlessly with another SOA Software offering, Repository Manager, to provide a comprehensive unified SOA Governance automation solution. Repository Manager provides SOA development governance for end-to-end software development lifecycle management across a wide range of SOA assets. Repository Manager is out-of-scope with COIL's focus for this project on SOA Operational Governance.

The interfaces between the distributed and centralized components use industry standards where available (UDDI, WS-MEX, WS-Policy, WS-Management, WS-Trust, etc).

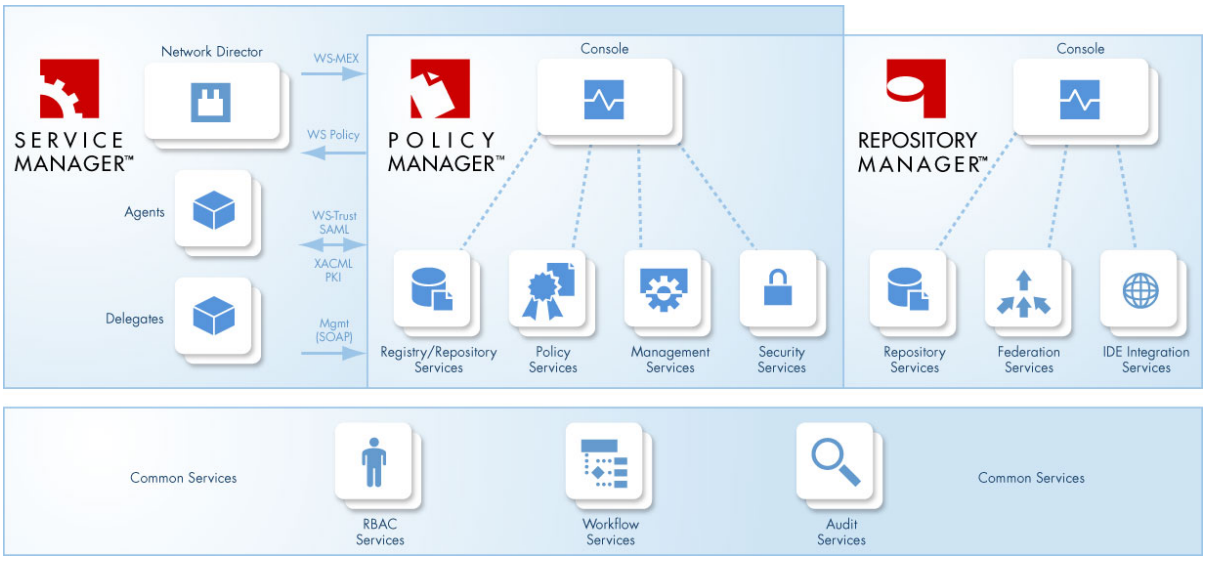


Figure 3 - Overview of SOA Software components

2.2.1 Service Manager

Service Manager provides the distributed intermediaries that implement and enforce policy for, and provide metrics and audit data back to, Policy Manager. It includes 3 distinct intermediary types:

- **Network Director**

  The Network Director is a stand-alone smart service router that deploys into the network supporting a wide range of intermediary patterns for routing, service virtualization, high-availability/load-balancing and others;

- **Agents** for most common service platforms and containers,

  The Agents deploy into service containers to ensure last-mile security and policy enforcement for services. SOA Software offers agents for most common Java application servers, .NET, several ESB products and several business process management tools.

- D**elegates**

  A delegate is a client-side intermediary that deploys into consumer applications to abstract the application from the location, transport, and policies required by the services it will consume. SOA Software offers delegates for Java and .NET applications

At COIL, the integration we tested is based on Network Director. Agents-based approach requires more tight integration with SAP NetWeaver Application Server and maybe included in future phases of the project.

2.2.2 Policy Manager

All of the central Policy Manager tools are implemented as standalone, stateless Java applications that leverage a central database for state management and as a core data repository. This delegates reliability and performance management to the underlying database tier, a well understood discipline in most large enterprises. SOA Software's products are designed to work with common database infrastructure solutions including Oracle, SQL Server, and DB2.

2.2.2.1 Web UI Console

Policy Manager provides a powerful web-based Console which is a JSR-168 compliance portlet-based application implemented in html, AJAX, and Flash. The Console delivers an integrated user interface for SOA registry/repository, policy management, governance, security, management, and monitoring.

The Console is a stateless Java Web application that deploys by default into its own self contained container. It can also be deployed into Tomcat, IBM WAS, or BEA WebLogic.

2.2.2.2 Registry/Repository Services

The cornerstone of Policy Manager is a set of registry/repository services that enable other Policy Manager and Service Manager applications and components to find and communicate with their peers, and to discover, enforce, and implement policies for governed services.

The registry/repository services provide multiple different interfaces into the same core set of data. Each of the interfaces will show a different subset of the data in different ways. The core interfaces are the UDDIv3 Inquiry, Publish, and Subscription APIs, WS-MetadataExchange, and REST. The UDDI APIs present and manage a structured view of the data defining and categorizing services. WS-MEX provides a standardized mechanism for retrieving WSDL, Policy, and other metadata, and the REST API provides an overarching mechanism for managing the complete data set.

2.2.2.3 Policy Services

Policy Manager's policy services extend the metadata repository described above with advanced policy authoring, governance, and distribution capabilities. They create and manage WS-Policy documents that describe the expected and required behavior of the services and service operations with which the policies are associated. They also create and manage lifecycle compliance policies that validate the static and dynamic metadata for a service. These policies include things like WS-I Basic Profile validation, service categorization checking, runtime policy presence validation, WSDL conformance, schema conformance and others. The set of policies delivered with the product can be readily extended using the XQuery language, or by adding custom modules created using a published and documented API.

In addition to their role managing policies, policy services define, negotiate, govern, and publish contracts. Contracts define the relationship between a service or group of services and a consumer or group of consumers. They are XML documents that define the access rights, capacity requirements, performance requirements, and other key contractual terms. Contracts are enforced dynamically by the Service Manager intermediaries.

2.2.2.4 Management Services

Policy Manager's management application monitors the performance, throughput, and usage of services and applications, and consolidates this information to provide valuable services such as SLA reporting, performance charting and trend analysis, and alert and exception management. The distributed intermediaries collect alert, performance, usage and message data according to the defined policies for each operation they manage, they then use SOAP and REST interfaces to push this data to the management application. The management application processes this captured data to calculate SLA performance, determine if any actions need to be taken, distribute alerts, and present real-time and historic charts.

The Policy Manager management application is based on the WS-Distributed Management specification, and will adopt whichever specifications emerge from the HP, Microsoft, and IBM sponsored harmonization initiative. It also supports SNMP and EIF for integrating with 3rd party management systems like HP OpenView, IBM Tivoli Enterprise Console, and CA Unicenter. It exposes a set of published, documented Web services APIs for easy integration with enterprise management portals.

2.2.2.5 Security Services

In the SOA Governance reference model, security services serve three purposes. They provide a token server, an authorization server, and a PKI certificate authority.

The security token server provides both authentication and token exchange services. It can consume a credential, and return a token of some description, most likely a SAML assertion in a Web services environment. A common use case for both authentication and token exchange in Web services is for the security token server to work in conjunction with a portal to request a username and password from a Web browser user, and provide the browser with an http session cookie. When the portal needs to request access to a Web service, it should then

contact the security token service and exchange the cookie for a SAML assertion. Policy Manager's security services support a wide range of token types and protocols including WS-Trust for requesting tokens, and SAML, Kerberos, WS-Security, HTTP Basic Authentication, HTTPS certificates, X.509, and others as token formats. They can delegate authentication decisions to external systems like Microsoft Active Directory, CA SiteMinder, and IBM TAM.

Policy Manager's security services provide an XACML compliant authorization server with a service for making decisions about whether a particular request is authorized or not based on a number of factors including user, role, or other sender identifying characteristics, request content, request destination, and environmental factors such as destination real-time performance. Most authorization servers still implement proprietary APIs, although XACML remains the most commonly discussed and implemented authorization standard. It can use external group information from systems like Microsoft Active Directory and LDAP servers, and can delegate authorization decisions to external systems like CA SiteMinder, and IBM TAM.

Policy Manager's security services also provide a built-in PKI solution with the ability to generate and manage public/private key pairs and certificates, import externally generated keys and certificates, and distribute these keys and certificates to the processes and applications that need them in real-time. The PKI solution supports certificate revocation list checking and uses an XKMS-based model for certificate and key distribution.

2.3 Network Integration of SOA Software and SAP components

One of the goals of the COIL exercise is to figure out a quick and easy way to integrate the SOA software solution without changing or adding a lot of footprint in existing SAP components. As we have established in section 2.2, unlike the agents which must be deployed to SAP application server, the SOA Software Network Directors (ND) sit in front of SAP application servers and behave like edge proxies that provide runtime enforcement points for Web services policies without any footprint in SAP application server. Essentially the ND proxies provide virtual Web service endpoints, and route web service requests, after applying policies and load balancing, to multiple physical web-service end points.

2.3.1 Recommended deployment pattern for SOA Operational Governance proxies

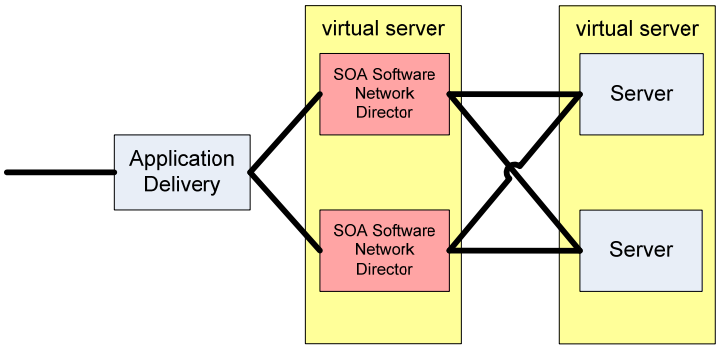


Figure 4 - Deployments of SOA Software Network Directors

As described in the Figure 4, for our landscape, we determined that the ND is the right solution and the best ND deployment pattern was to put them in between the AD and the application instances.

It should be noted, configuration was required on AD to load balance the traffic to the ND clusters that were configured in each of the respective environments. An ND cluster is a logical grouping of ND's. Services can be

hosted on the cluster, and transparently deployed to each of the ND's. WSDL's for services hosted on the cluster display the address of the load balancer (and not the ND).

With such deployment, the following was achieved:

- **Scalability:** SAP application component scalability can be achieved by adding as many application instances on as much server hardware as needed for some high throughput requirements. The same methodology is applicable for the NDs. With the load balancing capability of the ND's any ND can send traffic to any server as indicated by the "x" like connectivity lines in Figure 3. A configuration of m:n NDs:Servers with m,n being any integer number could be done.

- **High Availability:** with the important load balancing capability of the NDs and the "x" like connectivity between NDs and server instances, any single component failure would still allow continuous operation of the overall setup. Even some 2-point failures like one ND and one server failing simultaneously could be overcome.

2.3.2 Logical view of the integrated landscape

For the ND to function, the Policy Manager (PM) is also required. The COIL landscape also includes a NetApp file system storage which is made available on the local network as so called Storage Attached Network (SAN). During our tests with SOA Software, the SAN storage was used for storing logs.

With all the key elements included, the overall integrated landscape is described in figure 4.

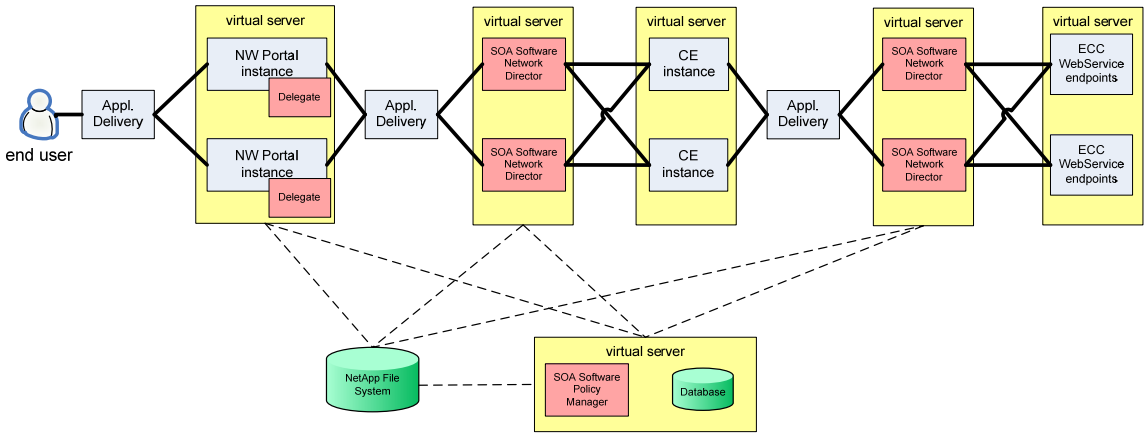


Figure 5 – The logical view of the integrated COIL and SOA Software landscape

For completeness, Figure 4 also shows another component in the SOA software stack – a client side Delegate which functions as an outbound policy enforcement point in the client side. The delegate is shown in the above picture but was not deployed during our testing with SAP portal.

Note that Figure 5 describes an ideal deployment of SOA software's SOA Operational Governance solution. Depending on the application scenario, it's not always necessary to deploy all the SOA components in the landscape. In the COIL scenario, the portal application requires dynamic WSIL and WSDL retrieval at run time. Our load testing scenario only deployed NDs for Web services traffic between CE and the ECC Backend. No ND is

deployed between SAP portal and CE. Also since our test application doesn't require advanced security policy enforcement, we didn't include the "Delegates" in our performance testing either.

3 Performance and stress testing

After deploying the SOA software components into the COIL landscape, we carried out a set of performance tests to assess the performance impact of the added components. Overall, the COIL tests showed very good results. The added response time by the SOA software proxies was minimal, and the stability in long-running stress tests was excellent.

Before we get into the details of the results, let's provide some background about the hardware configurations and policies enforced during the testing.

3.1 Hardware configurations of SOA components in COIL

To ease the installation process, we used VMware virtual images for the installation of SOA Software components.

- One virtual machine was set up to host Policy Manager and the Oracle database.

  Here is the configuration of the virtual machine: Disk Size: 16GB; Memory: 3GB; Operating System: Windows 2003 Standard Edition.

  SOA Software components installed in this virtual machine includes: Service Manager 5.2 with Update 1.0; Compliance Tech Preview Option Pack; Tracking Testing Tech Preview; and Oracle Database 10g

  Policy Manager contains 5 subsystems for dedicated functional areas, and each subsystem runs on a separate port. Out of the box default ports were used in the COIL landscape.

- Two virtual machines were set up to host the Network Directors.

  As shown in Figure 4, the Network Director virtual machine was replicated in the CE and ECC environments to provide redundancy and load balancing. Here is the detail for each of the machines: Disk Size: 16GB; Memory: 2 GB; Operating System: Windows 2003 Standard Edition

  SOA Software components installed are Service Manager 5.2 with Update 1.0, Compliance Tech Preview Option Pack, and Tracking Testing Tech Preview

  In the performance testing, only the virtual machine between CE and ECC were used.

3.2 Policy enforcement via SOA Software

As introduced in section 2, with SOA software solution, Web services policies are defined in Policy Manager and enforced at run time via either the Network Director or agents embedded into application servers. The Network Director provides policy enforcement for service providers without any footprint in SAP applications.

Often times, policies are defined based on the content or the sender of the messages. Different level of service level agreement (SLA) or business contract might be in place depending on which customer is being addressed. For example, for a gold-status customer, one may want to give higher priority in processing their orders. It's very important that an SOA Operational Governance solution is able to detect the sender or check the message content apply different policy accordingly.

In our Customer Fact Sheet scenario, we design the test scripts in a way that every 10^{th} call to the composite Web service the value of the customer Name field is set to be "badGuy". A policy is designed via Policy Manager and enforced via Network Director to check for "badGuy" in the messages, and if it's detected, an alert should be sent out and the whole message logged. Enforcement of such policy involves parsing the whole message which can be heavy processing especially when the message size is big.

The “badGuy” policy is enforced for some of the load testing we conducted.

3.3 Response time impact

In order to measure the response time impact for different size of web-service calls, the SAP backend component was configured to increase the response data set from call to call linearly. With the first call, the response message contains 10 sale order line items (message size around 30KB) ; second call with a response message containing 30 line items, so on so forth - with each of the following calls, the response message contains 20 more items. After 50 iterations, the response message is reset back to 10 line items, and the increasing cycle starts again. To measure the consistency of the impact, the test contains three cycles of increasing.

The COIL scenario only deployed NDs for Web services traffic between CE and the ECC Backend. No ND is deployed between SAP portal and CE.

In our test, HP Load Runner emulates users logging into the SAP portal, which calls web services on CE that in turn calls a set of ECC services to get customer basic data, quote history, and order history. Figure 6 shows the result of the message size ramping up test.

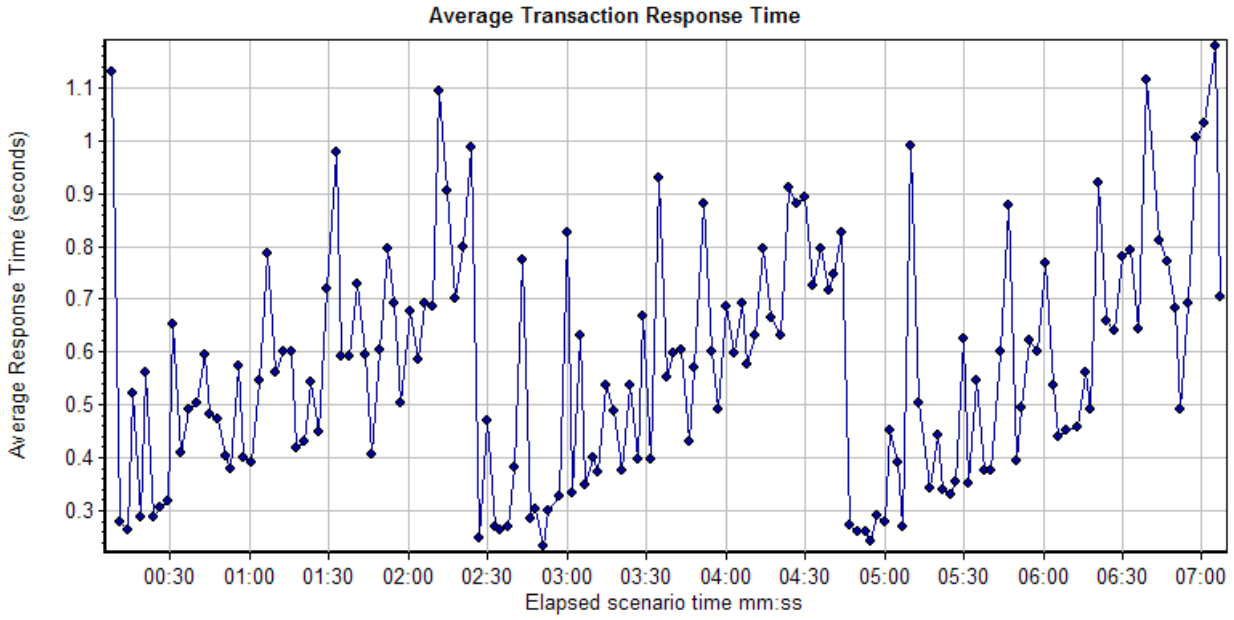


Figure 6 - Response time measurement from size ramp-up test

As can be seen in the result diagram, response time varies quite a lot from call to call, but follows a clear linear increase for the shortest response times.

It is not unusual for Java applications to occasionally have longer response times due to Garbage Collection short time holds of the Java VMs. In this measurement, there were synchronous calls from the Portal to the CE-Composite component to the ND and then to the ECC application and back. In total there are 8 hops through Java based components, increasing the likelihood of response time increases from Java Garbage Collection, in particular for calls with larger web-service content sizes. The low response times in Figure 6 are occasional calls where no Garbage collection occurred and therefore they follow the systematic linear increase as expected from a sequence of linearly in size growing web service calls.

It’s also important to understand how much overhead is added by the SOA Operational Governance component. The table below provides comparisons with different configurations of the ND component in front of the ECC component

Landscape	Response time (seconds)	
	Small XML	Large XML
SAP Baseline	0.24	0.42
+ ND (HTTP)	0.25	0.45
+ ND with badGuy Scan (HTTP)	0.25	0.6
+ ND with BadGuy Scan (HTTPS)	0.26	0.6

The response time listed above is what a Portal user would experience. The number is the sum of the response time of Portal, CE-Composite, ECC, and ND. The landscape variations compared are:

- SAP Baseline: Load balanced by F5, all HTTP traffic; No ND in place in the landscape.
- ND (HTTP): A ND is added in front of the ECC component. The F5 load balancer terminates https coming from the CE-Composite component and send http clear text traffic to the ND. The ND applies only light policies like statistic monitoring to the traffic.
- ND with BadGuy Scan (HTTP): Every 10th web service request was made to contain the string “badGuy” in the response content. An ND policy was deployed which detects the badGuy messages and logs them. This meant parsing and scanning the entire XML response content, which was considered heavy processing policy effort.
- ND with BadGuy Scan (HTTPS): in addition F5 was reconfigured to re-encrypt and route traffic with https to the ND. The ND had to terminate SSL.

The basic finding is that the ND is adding in the worst case 0.2 seconds response time with large messages (around 3MB). This is roughly comparable to the latency an SAP SOA component is adding when not executing business logic like in our test landscape. Compared to the response time fluctuations due to Java Garbage collection, the maximal 0.2 sec response time increase in our specific test landscape is likely no concern.

3.4 Stress Testing Results

In the stress test, first of all, over a period of a little more than 2 hours the number of users working in the NW Portal was linearly increased to 150 users, followed by a 5 hour period of constant high load testing. While the ramp-up phase demonstrates linear resource consumption proportional to the load, the high load phase checks for stability and reliability over time.

For all stress testing, the message size of web-service calls was kept constant. All the web service response messages contained 100 line items and are about 300KB in size.

This long running test was performed twice for 2 landscape configurations as described in the following sections.

3.4.1 Load Balanced with F5

In a load-balanced landscape, an F5 application delivery (AD) proxy is deployed in front of SAP Portal and CE clusters, and in front of the SOA Software ND cluster which sits before the ECC backend.

The F5 AD provides the following services:

- SSL termination: Incoming Client traffic was configured to be secured communication via https. The traffic from the F5 AD toward SAP servers or the NDs was then http. The SSL termination and decryption was delivered by the AD unit
- Load balance to the SAP components or to the ND Cluster

- TCP/IP connection multiplexing: The number of incoming client connection was channeled into much fewer server side connections, which frees server resources for connection handling.
- A few so-called iRules were deployed

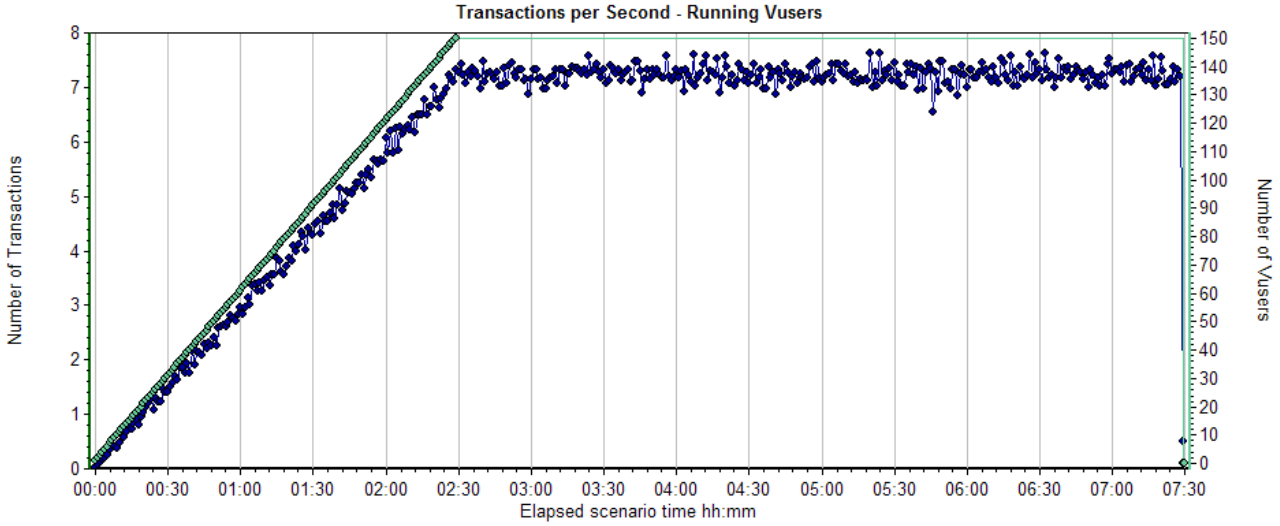
Further, the test script was designed such that the string “badGuy” occurred in every tenth transaction response. A policy is enforced in the ND’s to scan web service responses for “badGuy” which we considered as a heavy processing policy for the NDs because it meant full parsing and searching of the web-service responses.

As shown in the following table, our test found that the overall landscape with SOA Software components included was very stable and reliable.

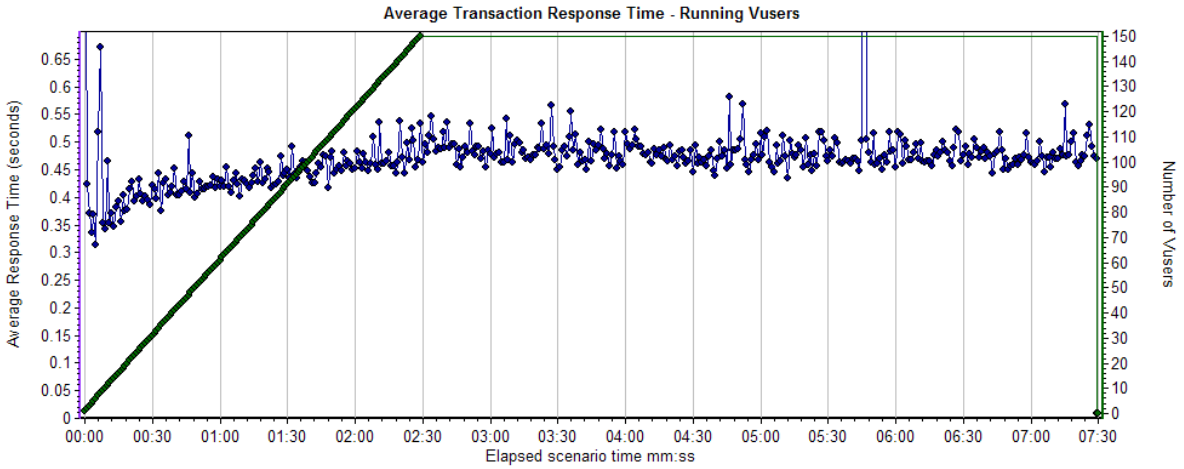
Response time (sec.) statistics					Error statistics		
Minimum	Average	Maximum	Std. Deviation	90 Percent	**Pass**	**Fail**	**Stop**
0.125	0.479	14	0.321	0.722	**162,440**	**0**	**0**

No error occurred within a total of more the 160,000 Load Runner transactions. This translates to a 5 nines availability of 99.9998%. SOA Software’s products clearly and impressively surpassed COIL’s standard pass criteria for network certifications of 99.9% error free.

The following diagram shows the transaction rate measurement. Here the measured transaction is a Load Runner transaction which covers the whole process starting with an emulated user clicking on a button in the SAP Portal application which causes multiple web service calls via ND to the ECC backend. As shown in the result diagram, the transaction rate follows perfectly linear the increase of users during the ramp-up phase and maintains a very constant rate during the 2nd high load endurance phase.



Now let’s have a look at how the increase of concurrent users impacts the response time. As shown in the following diagram, the response times increased slightly during the ramp-up phase and remained stable in the high load phase.



Another very important factor to check is how much resources are consumed by the SOA Operational Governance components. The following table shows the CPU usage of the deployed SOA Software components during the high load phase of the test.

	Network Director 1	Network Director 2	Policy Manager
CPU/%	7.7	8.2	7.2
network/KBps	430	430	33

Note that the NDs were installed inside VMware virtual machines and the CPU usage numbers are taken from the VMware monitoring tools.

3.4.2 Load Testing without F5

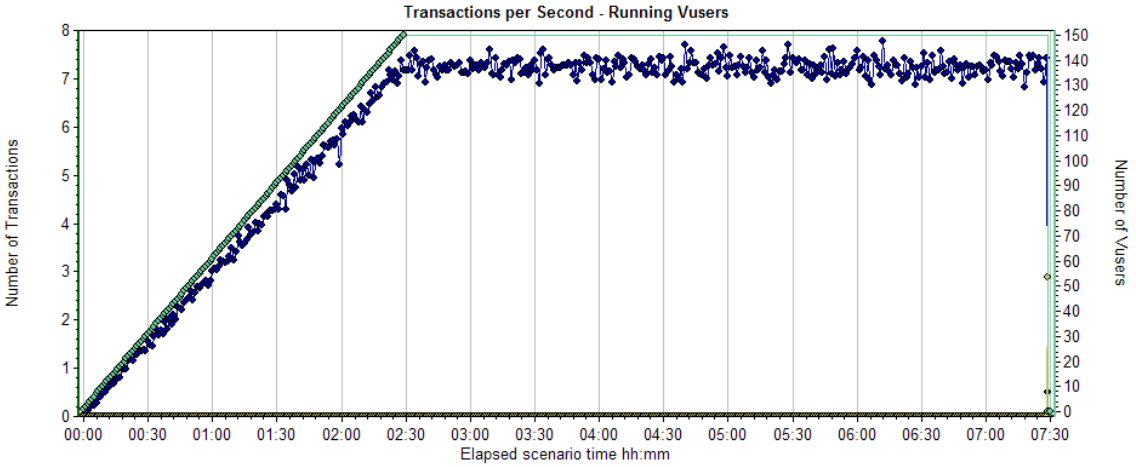
Since the F5 AD offloads a lot of processing from the NDs, we would like to evaluate how ND by itself performs without F5. We removed the AD in front of the backend NDs. Without the load balancing of F5, only one ND instance could be used in front of the SAP ECC backend. The ND instance took over the following functions:

- Terminating the SSL connection and traffic decryption.
- Handling more incoming client connections given that connection multiplexing feature offered by F5 is not available in this test.
- Load Balancing of traffic to the ECC backend instances

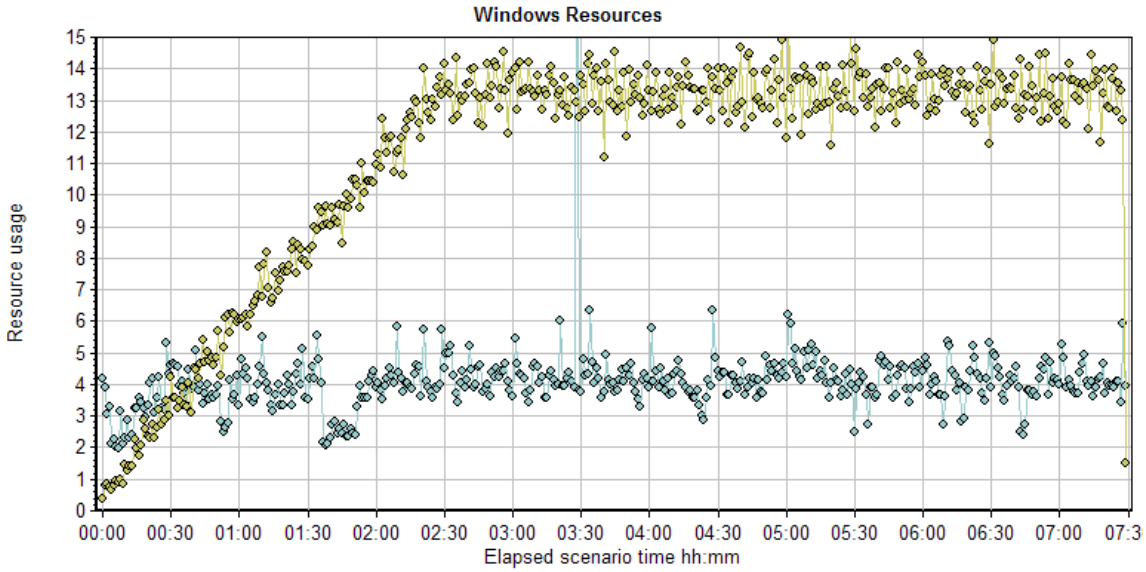
Since only one ND instance was used in this test, the ND instance has to handle twice the load than the other test described in section 3.4.1. Otherwise the test scenario was executed the same as in section 3.4.1.As shown in the following table, the overall result was equally excellent with regard to reliability and availability.

Response time (sec.) statistics					Error statistics		
Minimum	Average	Maximum	Std. Deviation	90 Percent	**Pass**	**Fail**	**Stop**
0.141	0.462	10.625	0.261	0.783	**162,761**	**0**	**0**

Similar linear load/resource-use scalability and good stability was also found in the transaction per second measurement.



In addition to the previous measurement we traced the CPU usage of the Network Director and the Policy Manager instances from inside the VMware virtual machine through the Windows performance monitor. The result is shown in the following diagram. The yellow line is the CPU usage of the ND and the blue line for Policy Manager. With the ND, again we found nicely linear CPU usage increase as load increases and stable CPU consumption during the endurance phase. Since only one ND was used instead of 2 as in the test before, CPU usage is about twice as high for this one node.



We noticed that the Policy Manager CPU usage is fairly constant during the ramp-up phase. Maybe the additional load is so small that it is not making a dent compared to the statistical fluctuations which are shown.

	Network Director	Policy Manager
Vmware CPU%	16	7.5
Windows CPU %	13.3	4.2
Diffence in %	16.9	44

A comparison of CPU usage measure with VMware tools vs. the windows monitoring values shows systematically higher values for the VMware measurements which likely can be attributed to the overhead needed for running in a virtualized environment. The bigger difference in the Policy Manager case might be due to the very low CPU usage of this component where measurement errors are relatively large.

4 Conclusion

Based on the testing performed at COIL with our specific landscape and the above described SOA Software components, we conclude that overall the COIL tests showed very good results.

- The added response time by the SOA software proxies was minimal.
- Stress tests revealed a 99.9998 error free operation.
- Resource usage scaled linearly with the applied load.
- The impact of handling secured https traffic vs. just http clear text traffic is minimal.

5 SOA Software Operational Governance Terminology

Term	Full Term	Description
ND	Network Director	Standalone java process, hosts, manages and secures virtual services at run time
PM	Policy Manager	Provides the configuration, design and governance user interface
SM	Service Manager	SOA Service Manager Subsystems (Registry Manager, Alert Manager, Policy Manager, Management Server). Each Service Manager subsystem runs as a separate java process and on its own port.
Container		Can be a standalone ND, an agent that runs natively in the application server or an ND cluster which is a logical grouping of standalone ND's
Contract		A contract is a Policy Manager object which describes the relationship between the consumer and provider.
Physical Service		The physical endpoint for a service. The physical service can be defined in Policy Manager from a WSDL URL, WSDL file, zip file, existing interface or can be defined without a WSDL.
Virtual Service		A physical service that is virtualized and hosted on an ND container. Once hosted, the virtual service can be managed, secured and monitored by Policy Manager.

6 References

SAP Co-Innovation Lab: https://www.sdn.sap.com/irj/sdn/coil

SOA Software: http://www.soa.com